0.11 — Configuring your compiler: Warning and error levels

ALEX[1] JANUARY 25, 2024

When you write your programs, the compiler will check to ensure you've followed the rules of the C++ language (assuming you've turned off compiler extensions, as per lesson 0.10 -- Configuring your compiler: Compiler extensions (https://www.learncpp.com/cpp-tutorial/configuring-your-compiler-compiler-extensions/)[2]).

If you have done something that definitively violates the rules of the language, the compiler is required to emit a **diagnostic message** (often called a **diagnostic** for short). The C++ standard does not define how diagnostic messages should be categorized or worded. However, there are some common conventions that compilers have adopted.

If compilation cannot continue due to the violation, then the compiler will emit an **error**, providing both line number containing the error, and some text about what was expected vs what was found. Errors stop the compilation from proceeding. The actual error may be on that line, or on a preceding line. Once you've identified and fixed the erroneous line(s) of code, you can try compiling again.

For advanced readers

Syntax errors are always diagnosed as errors.

If compilation can continue despite the violation, the compiler may decide to emit either an error or a **warning**. Warnings are similar to errors, but they do not halt compilation.

In some cases, the compiler may identify code that does not violate the rules of the language, but that it believes could be incorrect. In such cases, the compiler may decide to emit a warning as a notice to the programmer that something seems amiss.

Best practice

Don't let warnings pile up. Resolve them as you encounter them (as if they were errors). Otherwise a warning about a serious issue may be lost amongst warnings about non-serious issues.

In most cases, warnings can be resolved either by fixing the issue the warning is pointing out, or by rewriting the line of code generating the warning in such a way that the warning is no longer generated.

In rare cases, it may be necessary to explicitly tell the compiler to not generate a particular warning for the line of code in question. C++ does not support an official way to do this, but many individual compilers (including Visual Studio and GCC) offer solutions (via non-portable #pragma directives) to temporarily disable warnings.

Increasing your warning levels

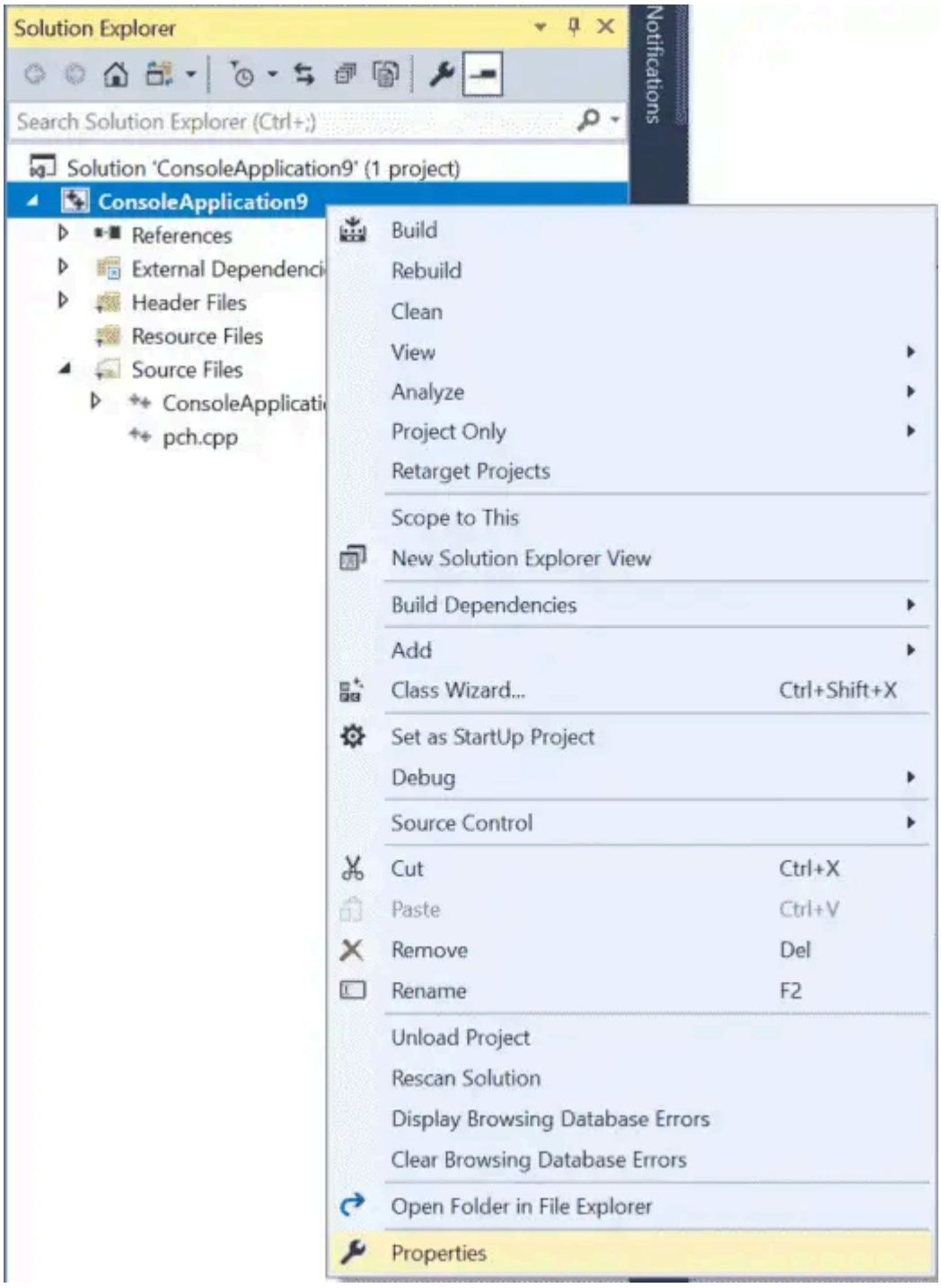
By default, most compilers will only generate warnings about the most obvious issues. However, you can request your compiler be more assertive about providing warnings for things it finds strange.

Best practice

Turn your warning levels up to the maximum, especially while you are learning. It will help you identify possible issues.

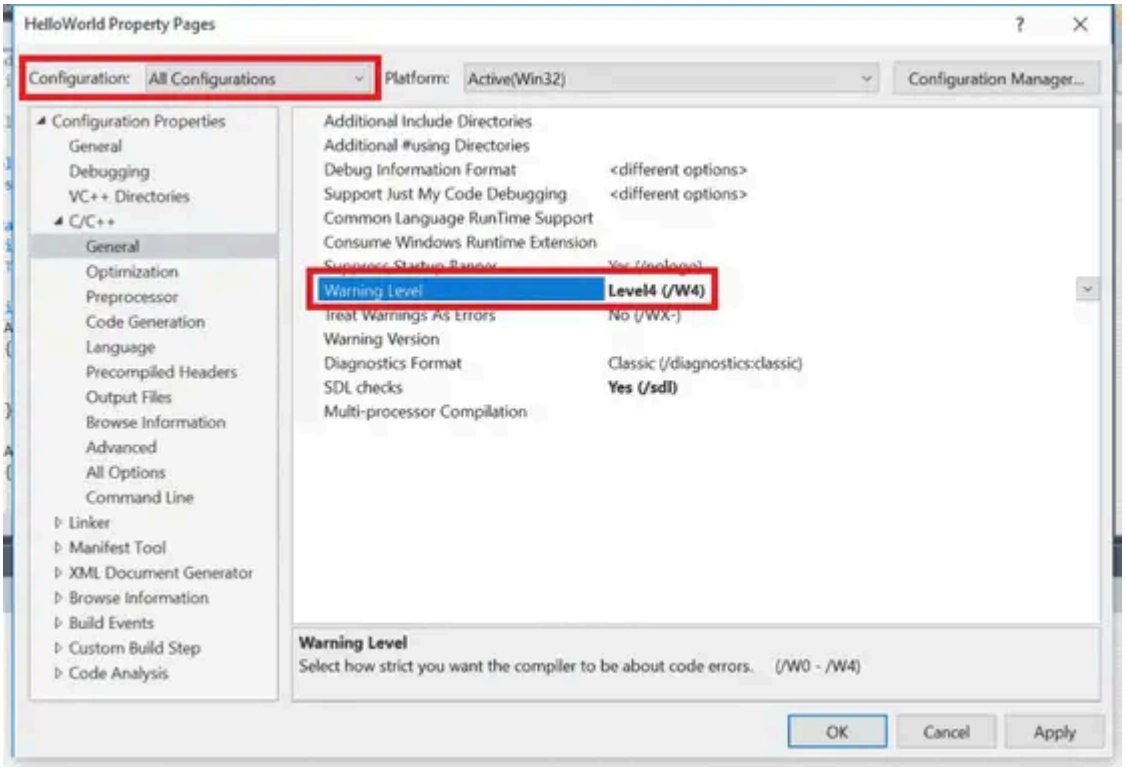
For Visual Studio users

To increase your warning levels, right click on your project name in the *Solution Explorer* window, then choose *Properties*:



From the *Project* dialog, first make sure the *Configuration* field is set to *All Configurations*.

Then select *C/C++ > General tab* and set *Warning level* to *Level4 (/W4)*:



Note: Do not choose *EnableAllWarnings (/Wall)* or you will be buried in warnings generated by the C++ standard library.

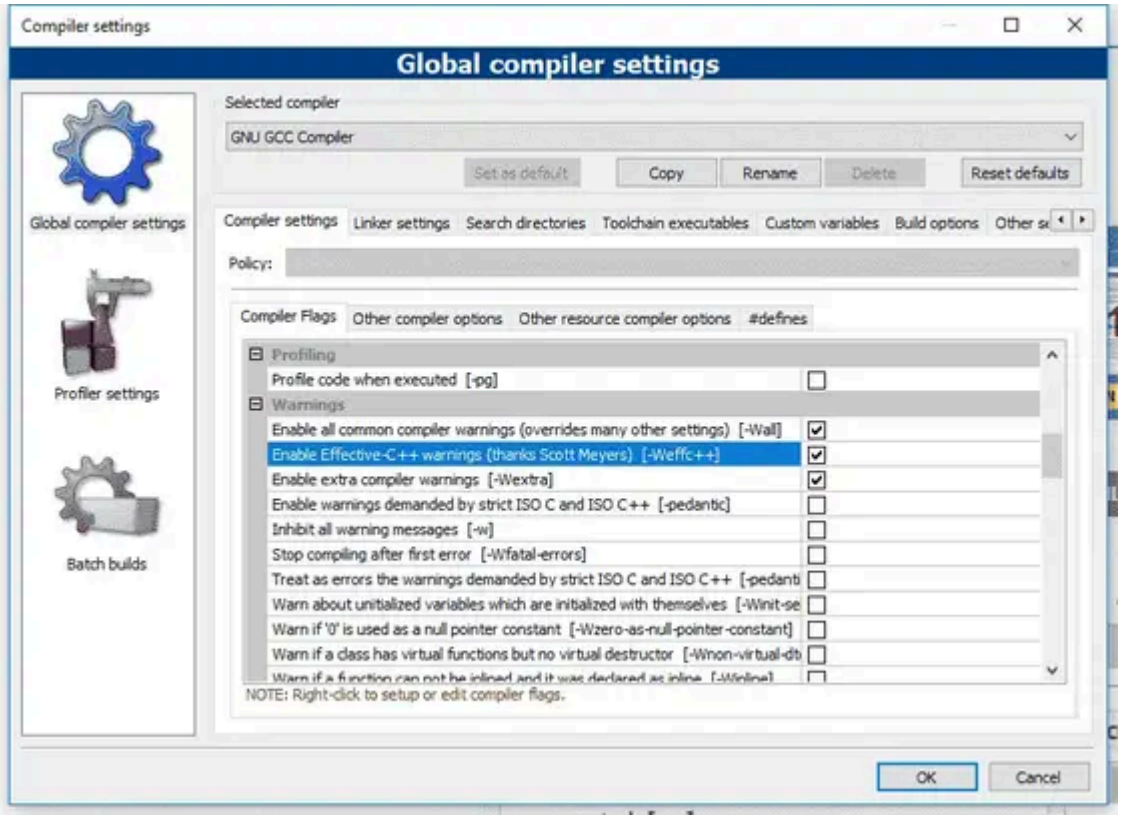
Visual Studio disables signed/unsigned conversion warnings by default, and those are useful, so if you are using Visual Studio 2019 or newer, let's enable those:

- From *C/C++ > Command Line tab*, under *Additional Options*, add `/w44365`. This tells the compiler to enable signed/unsigned conversion warnings at warning level 4 (which you enabled above).
- From *C/C++ > External Includes tab*, set *External Header Warning Level* to *Level3 (/external:W3)*. This tells the compiler to compile standard library headers at warning level 3 (instead of 4) so that compiling those headers doesn't trigger this warning.

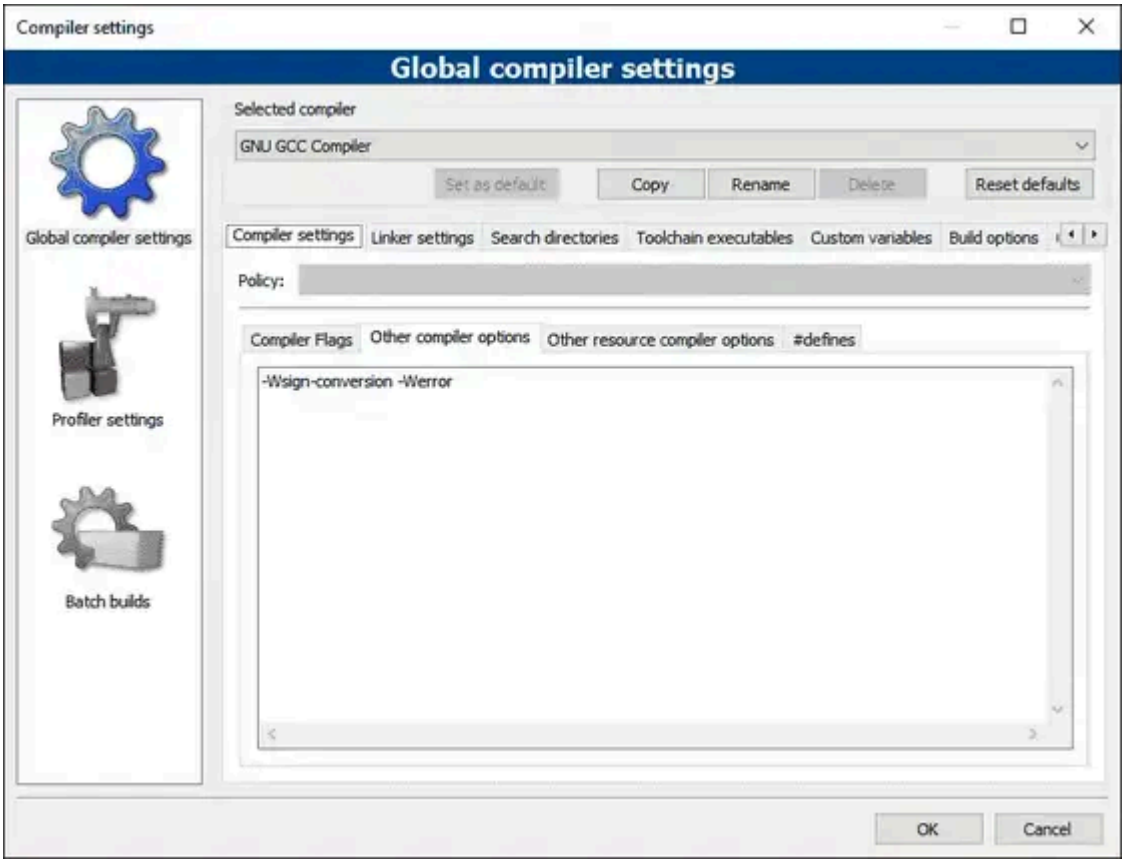
The "External Includes" tab isn't shown in the graphic above, but appears in VS Community 2019 or newer between the "Browse Information" and "Advanced" tabs. See this link (https://devblogs.microsoft.com/cppblog/customized-warning-levels-and-code-analysis-for-external-headers/)[4], which contains a recent photo of the dialog containing the "External Includes" tab.

For Code::Blocks users

From *Settings menu > Compiler > Compiler settings tab*, find and check the options that correlate with *-Wall*, *-Weffc++*, and *-Wextra*:



Then go to the *Other compiler options tab*, and add *-Wconversion -Wsign-conversion* to the following text edit area:



Note: The *-Werror* parameter is explained below.

For GCC/G++ users

Add the following flags to your command line: *-Wall -Weffc++ -Wextra -Wconversion -Wsign-conversion*

For VS Code users

Open the tasks.json file, find "args", and then locate the line *"${file}"* within that section.

Above the *"${file}"* line, add new lines containing the following commands (one per line):

```
"-Wall",
"-Weffc++",
"-Wextra",
"-Wconversion",
"-Wsign-conversion",
```

Treat warnings as errors

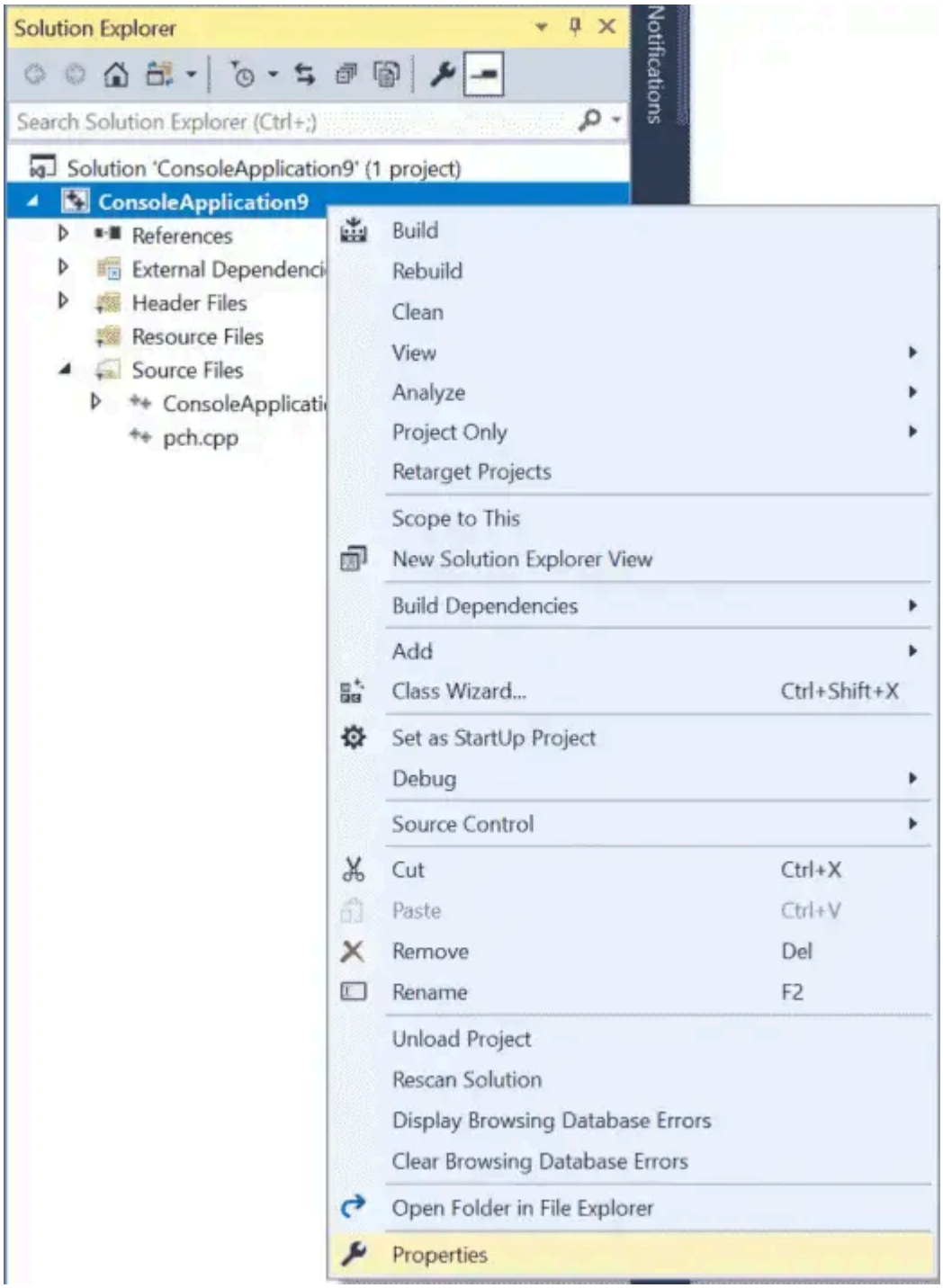
It is also possible to tell your compiler to treat all warnings as if they were errors (in which case, the compiler will halt compilation if it finds any warnings). This is a good way to enforce the recommendation that you should fix all warnings (if you lack self-discipline, which most of us do).

Best practice

Enable "Treat warnings as errors". This will force you to resolve all issues causing warnings.

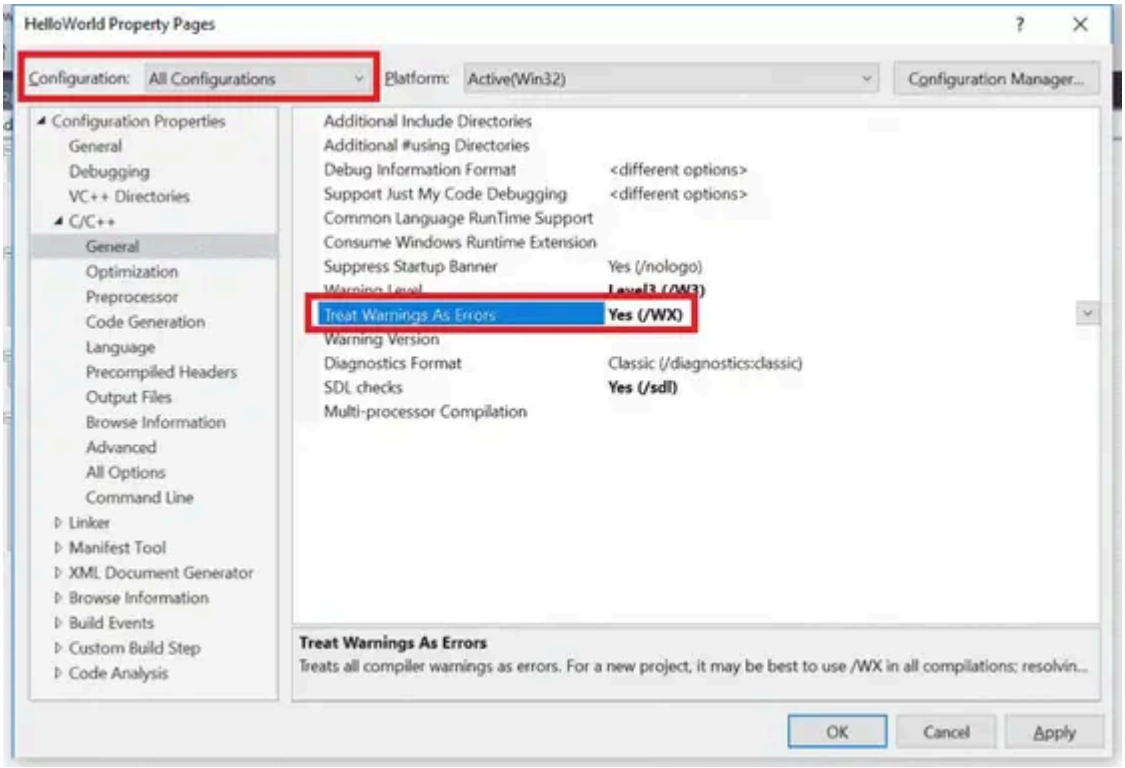
For Visual Studio users

To treat warnings as errors, right click on your project name in the *Solution Explorer* window, then choose *Properties*:



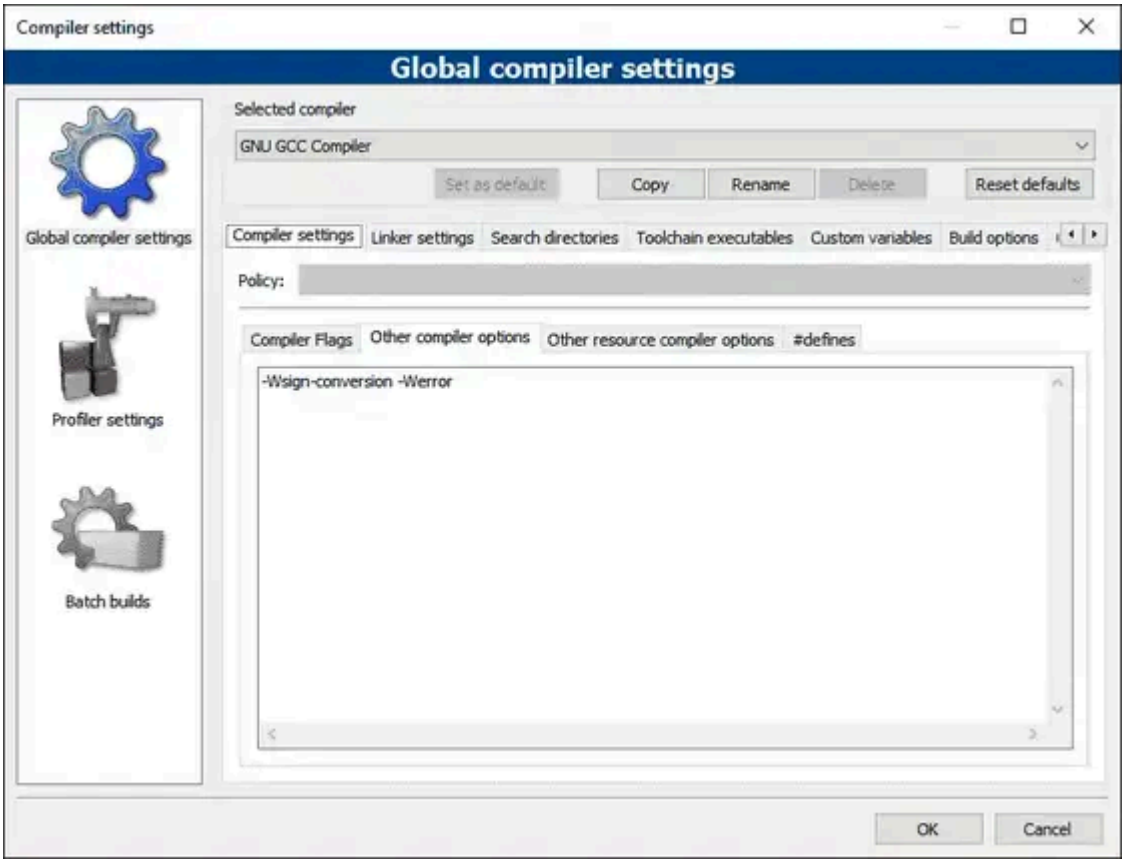
From the *Project* dialog, first make sure the *Configuration* field is set to *All Configurations*.

Then select *C/C++ > General tab* and set *Treat Warnings As Errors* to *Yes (/WX)*.



For Code::Blocks users

From *Settings menu > Compiler > Other compiler options tab*, add *-Werror* to the text edit area:



For GCC/G++ users

Add the following flag to your command line: *-Werror*

For VS Code users

In the `tasks.json` file, add the following flags before "${file}", one per line:

```
"-Werror",
```

Next lesson

0.12 Configuring your compiler: Choosing a language standard

5

Back to table of contents

6

Previous lesson

0.10 Configuring your compiler: Compiler extensions

2

7

B U URL INLINE CODE C++ CODE BLOCK HELP!

Leave a comment...

Name*

Email*

Notify me about replies:

POST COMMENT

Find a mistake? Leave a comment above!

Avatars from https://gravatar.com/[9] are connected to your provided email address.

177 COMMENTS Newest

Meerz
April 6, 2024 2:55 am

Does the "Treat warnings as errors" option only halts the compilation but still shows it as a warning, or does it make it look like an error ?

0 Reply

Alex Author
Reply to Meerz [10] April 6, 2024 11:37 am

It's up to the individual compiler.

0 Reply

Meerz
Reply to Alex [11] April 6, 2024 12:15 pm

Got it, thank you.

0 Reply

lollypoppy
March 15, 2024 8:04 pm

why are there 2 "For Code::Blocks users" parts?

0 Reply

Alex Author
Reply to lollypoppy [12] March 17, 2024 10:06 pm

The top on is for section "Increasing your warning levels", and the bottom one is for section "Treat warnings as errors"

0 Reply

riskatan
February 26, 2024 5:20 pm

For Xcode users:

Follow the first few steps of Rory's previous comment on Ch 0.10

1. In Xcode, press CMD+1 to show the Project Navigator.
2. In the Project Navigator click on your project, the one with the blue file icon.

3. In the main editor window, select the Target.
4. Click on Build Settings.
5. Make sure "All" and "Combined" are selected.

Then, to make all warnings appear as errors:

6. Type "Warning Policies" in the search filter.
7. Under "Treat warnings as errors" select "Yes"

Next, to increase the level of warnings:

8. Under "Pedantic Warnings" select "Yes"

I'm still very very new to all of this, but this is how I found the settings! Hope this is useful!

2 Reply

j3573r
February 22, 2024 8:58 am

hi, actually i did the compiler setup exactly as per your instructions but i am facing just one issue as of now while writing codes for cf contests. when i declare an array as:

```
1 | int arr[n];
```

it throws an error at me saying 'n' should be constant but i almost always ave to take it as input for various test cases. So i am unable to compile any such code in visual studio but it works completely fine in other compilers. Can you please let me know how to fix this or if i'll have to use a different way to declare an array or whatever the solution might be?

0 Reply

Alex Author
Reply to j3573r [13] February 23, 2024 2:44 pm

Arrays in C++ generally require the array length to be known at compile time.

If you need an array with a length that isn't known until runtime, use `std::vector`.

0 Reply

Swaminathan R
February 12, 2024 5:05 am

For some reason, I could only include -Wall in VS code, none of the others. I have MSVC compiler.

it errors out: cl : Command line error D8021 : invalid numeric argument '/Weffc++'

"-Weffc++",
"-Wextra",
"-Wconversion",
"-Wsign-conversion",

also,

"-Werror",

I included like this:
"args": [
"/Zi",
"/EHsc",
"/nologo",
"-Wall", //This works.
"-Weffc++",
"-Wextra",
"-Wconversion",
"-Wsign-conversion",
"-Werror",
"/Fe${workspaceFolder}\\task_mgr_app\\build_files\\usr1.exe",
"${workspaceFolder}\\task_mgr_app\\cpp_files*.cpp"
],

0 Reply

Scott G

Reply to Swaminathan R [14] April 7, 2024 12:53 pm

I am also having this issue. Please let me know if you've found a fix or know why it's happening.

0 Reply

Scott G

Reply to Scott G [15] April 7, 2024 2:22 pm

I figured out why it wasnt working for me. My .json was compiling using both the windows compiler (cl.exe) as well as the g++ compiler. The windows compiler was producing the error. Once I deleted the Windows compiler from my .json, it worked.

0 Reply

Swaminathan R

Reply to Scott G [16] April 8, 2024 12:20 am

Hello Mr. Scott, Thank you for your answer. Can you let me know the steps you followed to delete windows compiler?

0 Reply

Scott G

Reply to Swaminathan R [17] April 8, 2024 7:00 pm

in the launch.json file it should say "configurations" near the top, for me its on the second line. Following that line there will be at least one block of code surrounded by curly brackets. The block of code that contains the cl.exe text is telling the debugger to compile using the Windows compiler. I commented that block out, which left me with only the g++.exe compiler.

```
"configurations": [
// {
// "name": "C/C++: cl.exe build and debug active file",
// "type": "cppvsdbg",
// "request": "launch",
// "program": "${fileDirname}\\${fileBasenameNoExtension}.exe",
// "args": [],
// "stopAtEntry": false,
// "cwd": "${fileDirname}",
// "environment": [],
// "console": "integratedTerminal",
// "preLaunchTask": "C/C++: cl.exe build active file"
// },
{
"name": "C/C++: g++.exe build and debug active file",
"type": "cppdbg",
"request": "launch",
"program":
"${fileDirname}/build/Debug/\\${fileBasenameNoExtension}.exe",
"args": [],
"stopAtEntry": false,
"cwd": "${fileDirname}",
"environment": [],
"externalConsole": false,
"MIMode": "gdb",
"miDebuggerPath": "C:\\msys64\\ucrt64\\bin\\gdb.exe",
"setupCommands": [
{
"description": "Enable pretty-printing for gdb",
"text": "-enable-pretty-printing",
"ignoreFailures": true
},
```

```
{
"description": "Set Disassembly Flavor to Intel",
"text": "-gdb-set disassembly-flavor intel",
"ignoreFailures": true
}
],
"preLaunchTask": "C/C++: g++.exe build active file",
}
```

0 Reply

Scott G

Reply to Scott G [18] April 8, 2024 7:02 pm

I believe I also deleted the Windows compiler tasks from the tasks.json file, which follows similar steps. Both .json files are found in the .vscode directory.

0 Reply

IceFloe

February 7, 2024 5:53 pm

Of course, I use a web compiler, but I read the lessons on settings and functions carefully, I'm interested in how it all works, and even more so, I will return to this again as soon as the situation allows

0 Reply

De

February 4, 2024 12:00 pm

When trying to build a solution in VS studio, I get two errors,
The first one is an MSB4030 error that says :
Error MSB4030 "/WX" is an invalid value for the "TreatWarningAsError" parameter of the "CL" task. The "TreatWarningAsError" parameter is of type "System.Boolean"

And the second one : Element <WarningLevel> has an invalid value of "/W4".

I came back to this section to find a solution and I am sure I have selected the settings you reccomend exactly. I think resetting these property settings to default and reconfiguring them would help , but I don't know how to do that.

0 Reply

Den

Reply to De [19] February 9, 2024 9:26 am

Edit: I gave up on trying to figure it out and just created another project file, which works fine

0 Reply

ooomaxxx

January 5, 2024 11:59 am

In Xcode (15 in my case) to increase the warning levels, you'll want to enable more warning flags.

1. Open your Xcode project.
2. In the project navigator on the left, select the project file at the top of the list.
3. Select your target under the "Targets" section.
4. Go to the "Build Settings" tab.
5. In the search bar at the top right, type "Warning" to filter the settings related to warnings.
6. Look for the "Other Warning Flags" setting.
7. Double-click on the "Other Warning Flags" row to edit its value.
8. Add the warning flags that you want. For example, you can add `-Wall` to enable a set of commonly used warning flags. If you want to be more aggressive, you can use `-Weverything` to enable almost all warnings.

`-Wall`
or
`-Weverything`

Note: Using `-Weverything` may enable some very pedantic warnings that might not be suitable for all projects, so use it with caution.

9. Press Enter to save the changes.

To treat warnings as errors just type that in previous search bar and select "Yes", instead of default "No"

Last edited 3 months ago by ooomaxxx

3 Reply

AbeerOrTwo

December 13, 2023 6:59 pm

Need to figure out how to do this in Xcode but I won't sweat it too much

0 Reply

A. Leah

December 4, 2023 12:03 pm

Does anyone have recommendations on how to do this in Xcode?

0 Reply

Pasquale

Reply to A. Leah [20] December 13, 2023 3:09 am

i would like to know too

0 Reply

Links

1. https://www.learncpp.com/author/Alex/
2. https://www.learncpp.com/cpp-tutorial/configuring-your-compiler-compiler-extensions/
3. https://ikeamenu.com/humix/video/Ipms8sWsckf
4. https://devblogs.microsoft.com/cppblog/customized-warning-levels-and-code-analysis-for-external-headers/
5. https://www.learncpp.com/cpp-tutorial/configuring-your-compiler-choosing-a-language-standard/
6. https://www.learncpp.com/
7. https://www.learncpp.com/configuring-your-compiler-warning-and-error-levels/
8. https://www.learncpp.com/cpp-tutorial/variable-assignment-and-initialization/
9. https://gravatar.com/
10. https://www.learncpp.com/cpp-tutorial/configuring-your-compiler-warning-and-error-levels/#comment-595476
11. https://www.learncpp.com/cpp-tutorial/configuring-your-compiler-warning-and-error-levels/#comment-595485
12. https://www.learncpp.com/cpp-tutorial/configuring-your-compiler-warning-and-error-levels/#comment-594735
13. https://www.learncpp.com/cpp-tutorial/configuring-your-compiler-warning-and-error-levels/#comment-593911
14. https://www.learncpp.com/cpp-tutorial/configuring-your-compiler-warning-and-error-levels/#comment-593589
15. https://www.learncpp.com/cpp-tutorial/configuring-your-compiler-warning-and-error-levels/#comment-595515

16. https://www.learncpp.com/cpp-tutorial/configuring-your-compiler-warning-and-error-levels/#comment-595517
17. https://www.learncpp.com/cpp-tutorial/configuring-your-compiler-warning-and-error-levels/#comment-595522
18. https://www.learncpp.com/cpp-tutorial/configuring-your-compiler-warning-and-error-levels/#comment-595558
19. https://www.learncpp.com/cpp-tutorial/configuring-your-compiler-warning-and-error-levels/#comment-593238
20. https://www.learncpp.com/cpp-tutorial/configuring-your-compiler-warning-and-error-levels/#comment-590544